1

RADC-TR-89-328
Final Technical Report
January 1990

TACTICAL EXPERT MISSION PLANNER (TEMPLAR)

TRW Defense Systems Group

Sponsored by
Defense Advanced Research Projects Agency
ARPA Order No. 5769

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-89-328 has been reviewed and is approved for publication.

APPROVED: Albert G Frantz

ALBERT G. FRANTZ
Project Engineer

APPROVED: Raymond P. Urtz Jr

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: Igor G. Plonisch

IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COAD) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

TACTICAL EXPERT MISSION PLANNER (TEMPLAR)

Chuck Siska
Barry Press
Paul R. Lipinski

Contractor: TRW Defense Systems Group
Contract Number: F30602-85-C-0249
Effective Date of Contract: 3 September 1985
Contract Expiration Date: 3 December 1988
Short Title of Work: Tactical Expert Mission Planner (TEMPLAR)
Period of Work Covered: Sep 85 - Dec 88

Principal Investigator: Chuck Siska
Phone: (213) 812-8189

RADC Project Engineer: Albert G. Frantz
Phone: (315) 330-7764

Approved for public release; distribution unlimited.

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Albert G. Frantz, RADC (COAD), Griffiss AFB NY 13441-5700 under Contract F30602-85-C-0249.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
N/A

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
TEMPLAR CDRL A010

5. MONITORING ORGANIZATION REPORT NUMBER(S)
RADC-TR-89-328

6a. NAME OF PERFORMING ORGANIZATION
TRW Defense Systems Group

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
Rome Air Development Center (COAD)

6c. ADDRESS (City, State, and ZIP Code)
DH5/2461
1 Space Park
Redondo Beach CA 90278

7b. ADDRESS (City, State, and ZIP Code)
Griffiss AFB NY 13441-5700

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Defense Advanced Research Projects Agenc

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F30602-85-C-0249

8c. ADDRESS (City, State, and ZIP Code)
1400 Wilson Boulevard
Arlington VA 22209

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO
63789F	2321	05	04

11. TITLE (Include Security Classification)
TACTICAL EXPERT MISSION PLANNER (TEMPLAR)

12. PERSONAL AUTHOR(S)
Chuck Siska, Berry Press, Paul R. Lipinski

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM Sep 85 TO Dec 88

14. DATE OF REPORT (Year, Month, Day)
January 1990

15. PAGE COUNT
64

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	
25	05	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Knowledge based systems, Artificial intelligence,
Force Level Planning Systems, Object-oriented programming

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The TEMPLAR advanced development model was designed to prove Artificial Intelligence (AI) techniques could be applied to the large-scale Tactical Air Force problem of Air Tasking Order (ATO) generation. At over 300K lines of code, TEMPLAR is one of the large AI programs ever produced for the Air Force. TEMPLAR does detailed tracking and scheduling of aircraft and weapons use for an entire theater's Air Force assets. TEMPLAR runs on a Symbolics LISP machine and was written in LISP, Knowledge Craft and Flavors. Object-oriented forms and maps overlays present the information in a logical way to the users and connect to frames and demons that maintain the knowledge base. TEMPLAR allows for multiple, networked, planning workstations, each maintaining the entire current knowledge base at a corresponding communications cost.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Albert G. Frantz

22b. TELEPHONE (Include Area Code)
(315) 330-7764

22c. OFFICE SYMBOL
RADC (COAD)

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

TABLE OF CONTENTS

1.0 GENERAL

1.1 Purpose of Final Report

The objective of the Final Report on the TEMPLAR ADM development project is to report on the technical work accomplished, to identify the nature of the problems encountered in the course of completing the research, to report on both the positive and the negative results, and to report on the status of the testing.

1.1.1 Summary

Overall the TEMPLAR project is a success. TEMPLAR demonstrates that Artificial Intelligence techniques can be successfully applied to the Tactical Air Force problem of Air Tasking Order (ATO) generation. To date TEMPLAR stands as one if not the largest of the delivered AI programs in the Air Force. TEMPLAR is large in the sense of the objectives it met as well as the final size of the delivered code (almost 350K lines of LISP). There are three TEMPLAR systems installed, one at RADC (Rome Air Development Center), another at WPAFB (Wright Patterson Air Force Base) HRL (Human Resources Laboratory) and the third, after having been previously installed and demonstrated at HQ TAC(Tactical Air Command) for a year, is now at an operational user command, HQ 9th AF, Shaw AFB.

1.2 Project History

1.2.1 Overview

TEMPLAR was initiated to demonstrate that Artificial Intelligence techniques could be successfully applied to the full-scale operational problem of Air Tasking Order (ATO) generation. The TEMPLAR contract initiated in September of 1985 by the Air Force Rome Air Development Center (RADC) with TRW, followed the development of research prototypes at Mitre Corporations and an earlier TEMPLAR contract with another company. TEMPLAR also followed the development of the TAOTTS system at the Air Force Human Resources Laboratory (HRL). TAOTTS was never pursued separately following the start of TEMPLAR development.

At the time the project began, there was essentially no automation at Ninth Air Force (the contract-specified experts) to assist ATO planning. Two spreadsheets, PLANAID and TASKAID, had been developed to aid sortie tracking and tanker requirements planning. Both spreadsheets ran on personal computers. The output of those spreadsheets was able to be transferred to the Computer-Aided Force Management System (CAFMS), a system which essentially provides data base entry and processing services during the ATO development, via an interface between the personal computers and the CAFMS computers.

During the development of TEMPLAR, the spreadsheets were augmented with a BASIC program to assist in the flow planning for CAS and DCA missions.

Prior to the delivery of TEMPLAR, no comprehensive system[1] of automation for the Ninth Air Force ATO (9AF) planning process was available. Consequently, the ATO planning process, although described in a variety of Air Force publications and the subject of a number of prior contracts, was essentially a manual effort. The key operations, strategies, and methods of 9AF ATO planning had been developed over a number of years, but were largely undocumented. Many of the procedures were oriented around the idea that, by doing planning in a standardized way, combinatorial explosive numbers of planning variations could be avoided and the ATO completed in a reasonable amount of time.

1.2.2 Contract Phases

The TEMPLAR contract was organized into three phases. Phase one consisted of a review of prior technology, requirements development, and preparation of a design plan. Phase two was the development of the system. Phase three was integration, test, and analysis of the system. The contract called for one TEMPLAR system to be installed at RADC, with a period of performance of 33 months beginning September 1985. All external interfaces were to be simulated with the exception of the one for CAFMS. Contract modifications later added a number of requirements on the project, including the following:

- Delivery of TEMPLAR systems to HRL and to TAC Headquarters.
- Delivery of four incremental prototypes in addition to the final Advanced Development Model (ADM) software.
- Addition of a Limited ENemy Situation Correlation Element (LENSCE) electronic interface.

The TEMPLAR ADM software was installed and accepted at RADC in October of 1988.

1.2.3 Software Prototypes

The four TEMPLAR prototypes provided increasing functionality, leading to the delivery of the final ADM. The summary capabilities of the prototypes plus the ADM were these:

[1] It is recognized that the Air Force's CAFMS (Computer Assisted Force Management System) was a significant step in the automation of the TACC and generation of the ATO. It however, did not attempt to automate the portion of the planning process that TEMPLAR did.

1. Forms display and sequencing from one form to another. Rudimentary data entry capability, with no processing of entered data. Initial map display, with pan and zoom.
2. Initial implementation of selected forms. Initial map editing capability. Integration of Least Commitment Prototype planner for autoplanning demonstration capability. Initial scripting capability. Initial Natural Language input capability.
3. Initial functional Target Planning Worksheet, fuel calculations, tankering elements of Knowledge Base, and Guidance & Apportionment forms.
4. Conversion to Symbolics Genera 7.1. Full map editing capability. DMA videodisk map backgrounds. Additional forms implemented. Initial Natural Language input processing.

ADM. All functionality complete, including replacement of Least Commitment Planner prototype with package planner, flow planner, air-air refueling planner, and mission element planner. CAFMS and LENSCE interfaces complete to Interface Control Document specifications. Incorporation of classified Blue Flag scenario. Networking capability. Job Models. Knowledge Base editor. ATO save and restore. Enumeration, ordering and constraint checking.

1.2.4 Prior Automation

There is an important parallel in this chronology of software prototypes to the issue of delivering incremental prototypes. For the most part, the "AI" capabilities in TEMPLAR arrived in the final ADM delivery and were only present in demonstration form in the prototypes. The two key reasons for this were as follows:

- The effort during the first year of the program was devoted to documentation that was required by the contract and dictated by the conventional software development project schedule. The workload in preparing the document deliverables up through August of 1986, culminating with the draft of the Functional Description (FD), was such that the project had no time for large-scale prototyping efforts to validate the Knowledge Base design and AI algorithms. When combined with the fluidity of the design at the time the FD was published, this lack of time to validate the Knowledge Base and the AI algorithms can be used as the basis for a compelling argument that *conventional documentation schedules are not right for AI systems development.*
- The fact that there was no initial automation base for the ATO planning process cannot be overemphasized in importance. Before the Knowledge Base and approaches for the AI planners could be solidified, it was necessary to produce and validate a manual, non-AI planning system. That non-AI planning system

was the essential content of system version 4.8, which was completed after the formal delivery of prototype 4. Most of the Knowledge Base used for the AI planners was complete at the time version 4.8 was released.

Conventional automation could have been valuable in the development of TEMPLAR in other ways:

- o Sophisticated fuel consumption calculation software was believed to be available from the TAOTTS development. The completeness and accuracy of the TAOTTS code remains unclear. Testing revealed enough problems that the project decided that it would be impractical to use it as a base for TEMPLAR fuel calculations. Several attempts were then made with the Air Force to acquire fuel calculation software developed organically. Problems of differing hardware/software systems and diffuse points of contact made this impractical as well. The final fuel calculation software used in TEMPLAR is algorithmically derived from the simplified fuel calculations used in the PLANAID spreadsheets since these calculations are known to be compatible with other knowledge being acquired from 9AF.
- o Communications interface definition was a continuous problem. Issues addressed but unresolved included definition of which system version would be the baseline, acquiring interface specification data, and acquiring interface test data.

Neither of these (or several other issues) are unique to AI systems. The existence of prior operational automation of the ATO planning process could have served to clarify or solve these problems and to simplify the TEMPLAR development.

1.2.5 Deliverables

TEMPLAR produced the following deliverables in addition to this Final Report:

User's Manual

Maintenance Manual

Test Plan

Interface Control Document

Functional Description

Data Base Specification

Interim Technical Report

Design Plan

See the next section, 1.3 Project References, for complete references for these documents.

1.3 Project References

Following are the documents applicable to the history and development of the TEMPLAR project and are listed for information purposes only. The primary documents are the Interim Technical Report, Design Plan, Functional Description, User's Manual and Data Base Specification by TRW.

TEMPLAR (Tactical Expert Mission Planner), was developed by TRW Defense Systems Group for the U.S. Air Force, Rome Air Development Center (RADC). TEMPLAR is the first major Air Force application of Artificial Intelligence (AI) technology to produce an Advanced Development Model (ADM) of a decision aid which will be demonstrated by the operational forces. TEMPLAR builds on technologies demonstrated by RADC, the Mitre Corporation and the Defense Advanced Research Projects Agency (DARPA) to incorporate natural language understanding, constraint and expert system based planning, and advanced man-machine interface (MMI) techniques.

The TEMPLAR project sponsor was the Air Force Rome Air Development Center (RADC). The end users will be mission planners from the Air Force Tactical Air Control Centers (TACC). The system was installed at the Battle Management Laboratory at RADC, Griffiss AFB, NY, the Ninth Air Force at Shaw AFB, SC, and the Human Resources Laboratory (HRL) at Wright-Patterson AFB, OH.

a. Project Request

Tactical Expert Mission Planner (TEMPLAR), RFP F30602-85-R-0104, issued 10 April 1985.

b. Technical Documentation

"User's Manual (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL A007, by TRW Defense Systems Group, dated November 9, 1988.

"Maintenance Manual (Final), Tactical Expert Mission Planner (TEMFLAR)", CDRL A012, by TRW Defense Systems Group, dated October 1988.

"Test Plan (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL

A009, by TRW Defense Systems Group, dated August 1, 1988.

"Interface Control Document (Final), Tactical Expert Mission Planner (TEMPLAR)", by TRW Defense Systems Group, dated June 1, 1988.

"Functional Description (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL A006, by TRW Defense Systems Group, dated 18 March 1987.

"Data Base Specification (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL A011, by TRW Defense Systems Group, dated October 1987.

"Interim Technical Report (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL A005, by TRW Defense Systems Group, dated 15 October 1986.

"Design Plan (Final), Tactical Expert Mission Planner (TEMPLAR)", CDRL A004, by TRW Defense Systems Group, dated 3 July 1986.

"Tactical Air Force Headquarters and Tactical Air Control Center", TAC Regulation 55-45, dated 26 October 1984.

"TEMPLAR Design", RADC-TR-84-134, by the Advanced Information and Decision Systems (ADS), dated June 1984.

"Headquarters/Tactical Air Control Center USCENTAF", Regulation 55-45, dated 20 December 1985.

"Tactical Air Intelligence Handbook", TAC Manual 200-10, dated April 1969.

"An Introduction to Air Force Targeting", AF Pamphlet 200-17, dated 11 October 1968.

"Aerospace Operational Doctrine, Tactical Air Operations - Counter Air, Close Air Support, and Air Interdiction Handbook," USAF, dated 2 May 1969.

c. Documentation concerning related projects.

"KNOBS -- The Final Report (1982)", R. H. Brown, J. K. Millen, and E. A. Scarl; RADC-TR-86-95, August 1986.

"KNOBS Architecture", J.K. Millen, C. Engleman, E.A. Scarl and M.J. Pazzani;

(Draft), no date.

"The KNOBS System", E.A. Scarl, C.Engleman, M.J. Pazzani, J. Millen, to appear in a book edited by W. Zachary, ~1985.

"KNOBS: An Integrated AI Interactive Planning Architecture", C. Engleman, J.K. Millen, E.A. Scarl; AIAA Computers in Aerospace Conference, Hartford, Conn.

"MITRE/INTERLISP FRL Reference Manual", (Draft), dated December 1979.

d. Other manuals or documents that constrain or explain technical factors affecting project development.

"Constraint-Directed Search: A Case of Job-Shop Scheduling", Carnegie Mellon University publication CMU-RI-TR-83-22, by Mark S. Fox, dated 13 December 1983.

e. Standard or Reference documentation

(1) Programming conventions.

"RADC Computer Software Development Specification General Specification for ", Specification No. CP 0787796100E, dated 30 May 1979 and Amendment No. 1 to CP 0787796100E, dated 14 April 1981.

(2) DoD or Federal standards.

"Automated Data Systems (ADS) Documentation Standards", DoD 7935.1-STD, dated April 24, 1984.

"Economic Analysis and Program Evaluation for Resource Management", Department of Defense Instruction, Number 7041.3, dated October 18, 1972.

(3) Hardware Manuals and System Support Documentation

Knowledge Craft Manual Guide, Version 3.1, Carnegie Group Inc., 2 vols, dated 18 August 1986.

Knowledge Craft Installation Notes and Release Notes, Carnegie Group Inc., Version 3.1 for Symbolics 7.0/7.1, dated 17 July 1987.

Language Craft Reference Manual, Version 3.1, Carnegie Group Inc., dated 7 December 1987.

Symbolics Manuals, Release 7.0/7.2, Symbolics Corp., vols 0 to 10, dated June, July, August 1986, February 1988.

Symbolics Software Services Technical Bulletin, Genera 7.2, dated June 1988.

"ULTRIX-32 Programmer's Manual", Digital Equipment Corporation, Binders I, II, IIIA & IIIB, dated May 1984.

"ULTRIX-32 Supplementary Documentation", Digital Equipment Corporation, Vols, I - III, dated May 1984.

1.4 Terms and Abbreviations

AA	AntiAircraft
AAR	Air to AIr Refueling
ADM	Advanced Development model
AI	Air Interdiction
AIRDEF	Air Defense
ARCT	Air Refueling Contact Time
ATO	Air Tasking Order
AWACS	Airborne Warning and Control System
BAI	Battlefield Air Interdiction
BE	Basic Encyclopedia
CAD	Computer Aided Design
CAFMS	Computer Assisted Force Management System
CAS	Close Air Support
CDRL	Contract Data Requirements List
COMUSCENTAF	Commander US Central Air Force
COPS	Combat Operations Planning
DCA	Defensive Counter Air
DOD	Department of Defense
DS	Defense Suppression
EC	Electronic Combat
EOB	Enemy Order of Battle *or* Electronic Order of Battle
ETOT	Estimated Time On Target
FEBA	Forward Edge of Battle Area

FEP	Front End Processor
FLOT	Forward Line Of Troops
FOB	Friendly Order of Battle also form object
FP	Force Protection
FSCL	Fire Support Coordination Line
G&A	Guidance & Allocation
HEL	Helicopter
HWY	Highway
ICD	Interface Control Document
INT	Interdiction
KB	Knowledge Base
LENSCE	Limited Enemy Situation Correlation Element
LMFS	Lisp Machine File System
MB	MegaByte
MIL	Military
MMI	Man Machine Interface
NL	Natural Language
NTC	Night Targeting Cell
OB	Order of Battle
OCA	Offensive Counter Air
OPNS	Operations
PAA	Possessed Aircraft Authorized
PCC	Package Checker Cell
PD	Probability of Destruction
POL	Petroleum, Oil, Lubricants
RADC	Rome Air Development Center
REC	Reconnaissance
RWY	Railway
SCL	Standard Configured Load
SEAD	Suppression of Enemy Air Defense

SIF	Selective Identification Feature
TAC	Tactical Air Command
TEMPLAR	Tactical Expert Mission Planner
TFT	Time off Target
TOC	Table of Contents also Tactical Operations Center
TOT	Time On Target
TPW	Target Planning Worksheet
UM	User's Manual
USAF	United States Air Force

2.0 SYSTEM IMPLEMENTATION

2.1 Mixed Initiative

TEMPLAR had a requirement derived from KNOBS to support interleaved planning by the user and the system, a process called mixed initiative planning. In contrast to batch-like processing, mixed initiative planning gives the user control over the scope of what is to be autoplanned and, more importantly, gives the user the option to tune the intermediate results of the autoplanner before invoking the planner again.

Mixed initiative planning was successful in TEMPLAR for the same reasons as in KNOBS: TEMPLAR conducts planning by filling in forms, and it matters not to the planning process whether the user or the autoplanner fills in the form. By designing the autoplanner to distinguish data which are user-entered (and therefore sacrosanct[2]) from those which are autoplanner-generated (and therefore fair game), any number of iterations of the planning may be initiated.

Fields on forms in TEMPLAR map to slots in frames -- that is, every field on every form is attached to exactly one slot in one frame[3]. TEMPLAR retains knowledge of what has been filled in by the user (vs. the autoplanner) by marking frames as user-owned or autoplanner-owned; all slots in a frame so marked acquire that property.

This notion of ownership marking at a higher level than the slot reflects the concept that the value in a slot is only valid in some larger context. For example, an SCL of 4M84 is only valid for a certain combination of mission parameters including aircraft type, target, etc. In the design of the TEMPLAR Knowledge Base, such related sets of slots were grouped into a frame of a class reflecting that context, such as MISSION. Marking the entire mission as user-owned when any slot in the mission is filled in by the user, and clearing user ownership only when all slots in the frame are cleared, captures the idea of user or autoplanner ownership of the complete set of related mission data.

This idea of grouping slots into frames reflecting user-visible concepts imposes a constraint on the designer of the Knowledge Base. Since the groupings of slots into frames

[2] Sacrosanct is used here in the context of sacred or inviolable.

[3] Very loosely speaking, a frame is analogous to a record in a data base and a slot is analogous to a field in a record, although the analogy fails to express the additional power provided by frames and slots relative to traditional data bases. To illustrate frames and slots, in TEMPLAR a common frame is one to represent a single mission. Slots in that frame contain data defining the unit, aircraft type, number of aircraft, SCL, call sign, mission number, owning package, and many other items.

determines the scope of what the user marks as user-owned by entering data, the designer becomes constrained to organize the Knowledge Base such that the resulting groupings are useful from both a user and a TEMPLAR implementation perspective. In practice, this was never a problem -- common frame classes in TEMPLAR include missions, Target Planning Worksheets, and other conceptual objects which are both familiar to the user and of a level manipulated by the autoplanner. So close is this correspondence that the TEMPLAR mechanisms for forms implementation is able to rely upon the equivalence of frames and user-visible groupings to implement repeated form structures and most of the key field-to-slot linkages.

There are four distinct planning modules in TEMPLAR:[4]

- o The package planner, which generates mission lines on Target Planning Worksheets.
- o The flow planner, which generates mission lines on unit schedules for CAS and DCA missions.
- o The air-air refueling planner, which generates mission lines on Refueling Planning Worksheets.
- o The mission line planner, which fills in mission number, call sign, SIF code, and other information on existing mission lines.

Because of the diversity in the design of these four planning mechanisms, we decided to make the recognition of which slots hold user-owned values and which slots were free for autoplanning the responsibility of each individual planning module. As a result, recognition of user-owned slots is programmed into the planners (and somewhat *ad hoc)* rather than automatic or the result of a broad architectural decision. This decision turned out to be correct, however, since the ways in which the planners deal with user-owned information differs considerably.

- o The best integration of user data and automatic planning is in the package planner during its depth-first tree walk for a single Target Planning Worksheet. User-entered data is taken as the only enumerable values during the search. In addition, when such information is available it can be used to guide the candidate value enumeration functions for related slots.
- o This level of user/autoplanner integration was not possible in the other planners,

[4] In TEMPLAR, a mission is a set of sorties all going to the same place at the same time with a common objective. A package in the current implementation of TEMPLAR follows Ninth Air Force practice in that it consists of attack missions against one or more targets combined with support missions as required for force protection, defense suppression, electronic combat, and reconnaissance.

which are more algorithmic in nature than the classic search used in the package planner. For example, the flow planner attempts to create a set of missions meeting a set of specifications detailing aircraft types, numbers, TOT's, inter-arrival times, etc. Incorporating user data into the flow plan implies adjusting the flow to correspond to user-entered missions. The process of effectively allowing for small variations in the user missions to still map into the flow plan became complex, and so the implemented flow planner ignores (but does not alter) missions planned by the user. Similar problems in the air-air refueling planner caused its design to plan around user-entered fuel requirements or tanker missions rather than to attempt to integrate the user information tightly into the tanker plan.

Mixed initiative planning is also assisted by TEMPLAR's provision of multiple levels of initiation for the various autoplanner functions, ranging from planning one field through planning one or more frames (e.g., mission lines, Target Planning Worksheets, etc.), to planning the entire ATO. By providing the user with a finer degree of control than all-or-nothing, it is possible to use the planner in only those situations in which the user believes the system can contribute effectively.

The air-air refueling planner is an exception to this control scheme; the air-air planner operates on an all-or-nothing basis. Additional algorithm work in that planner would enable it to have the same degrees of control as the other autoplanner components. The SOW specifically prohibited looking at replanning, making changes to these restrictions out of scope of the TEMPLAR contract.

Giving TEMPLAR the power to handle the entire ATO planning process did require some impositions to be enforced. In particular, because changes made to setup data can invalidate elements of the Knowledge Base setup during planning, TEMPLAR imposes some restrictions on the order of events during planning. Essentially, these restrictions require that certain scenario elements be setup before other setup data, and that all setup data be complete before planning begins. Some of these restrictions may be overly rigorous, such as precluding movement of tanker tracks once planning is started. A review of the system could verify that some restrictions can be removed; demons can be provided to remove others (such as the one on tanker locations).

2.2 Enumeration and Ordering

TEMPLAR augments the model of planning via filling out forms with active assistance to the user. Included in the active assistance features are functions to provide enumeration of values which may be entered into a field, and functions which will provide ordered recommendations for those values.

As implemented, TEMPLAR extends the notion of enumeration to include both blind enumeration and intelligent enumeration.

- Blind enumeration is driven off of the name of the slot being filled in, and is available nearly everywhere in the system. Internal functions attached to the slot names are capable of returning either all the values which may be entered or, in the case of fields with continuous ranges of values, suggested candidate values. Blind enumeration is independent of the local or global context of the slot, however, and so can end up presenting candidates which are invalid in context.
- Intelligent enumeration works by taking the blind enumeration values and processing each of them through the Formal Constraint Language constraints attached to the slot. For certain slots, specialized inversions of constraints directly implemented in LISP are also used. The surviving values presented to the user are known to be valid in the context. The process takes longer than blind enumeration, but produces better results.
- Ordering works by passing each of the values from intelligent enumeration through the Formal Constraint Language preferences attached to the slot. The resulting ordered set is presented to the user. In the current system, the ordering process is not significantly slower than intelligent enumeration.

All three of these mechanisms appear to be valuable to users, and to be applicable across the range of TEMPLAR planning functions. There are limitations on the availability of intelligent enumeration and of ordering, simply because constraints and preferences in the Formal Constraint Language are not available everywhere in the system. Further effort to define such additional constraints and preferences would extend the availability of these functions, and would simultaneously improve the intelligence of the planning functions driven off of the Formal Constraint Language.

The desirability of extending the delivered ADM to include more constraints and preferences highlights one of the characteristics of TEMPLAR which we believe applies to most operational-scope AI systems. Because the problem TEMPLAR addresses is so broad and complex, it is not possible to do concrete, in-depth engineering of the Knowledge Base with the users until nearly all of the manual-planning mode system functionality is available. In the TEMPLAR development, this did not occur until approximately 6 months before final delivery of the software, limiting the volume of knowledge that could be inserted via the constraints and preferences. This bottleneck can be addressed in at least two ways:

1. There should be a period of system evaluation with the users following delivery

of the completed software[5]. This period should include support by the software development team, and should at least have these objectives:

- o Expanding and refining the system Knowledge Base.
- o Training operational users in the use and administration of the system.
- o Fixing problems uncovered under operational load not evident during development test.
- o Tuning system speed in areas which present problems for operational use.

2. The normal development schedule for documentation and software prototypes should be re-evaluated for operational AI systems. In particular, the TEMPLAR program emphasized too much documentation early in the program when concepts and ideas were too fuzzy to be worth documenting extensively. As a result software prototyping to clarify those ideas did not start until later than would have been desirable.

2.3 Automatic Checking of Evolving Plans

TEMPLAR provides automatic checking of evolving plans by examining each value typed into a field on a form. A wide range of mechanisms accomplishes these checks:

- o Syntactic checkers examine each keystroke to determine if the input contributes to a syntactically legal input. Example: Times must be in HHMM format, with the ranges for hours and minutes bounded appropriately.
- o Semantic checkers determine if the complete entry makes sense in the context of the Knowledge Base. Example: An SCL name must correspond to one of the SCL's known to the system. If the entered values does not match a known SCL, the system queries the user whether or not the entered name should be added to the set of known SCL's.
- o Demons, which in this case are procedures triggered when a value is stored into the Knowledge Base, check for and attempt to maintain internal consistency of the Knowledge Base. Example: Demons are responsible for maintaining the internal record of what airframes are allocated to which missions at what times, and for notifying the user when a given subunit has been overtasked.
- o Constraints written in the Formal Constraint Language evaluate the input value for compliance with restrictions imposed by good planning practice. For

[5] To build an operational AI system as compared to an ADM there would also be a need for development and operation test and evaluation.

example, a constraint in the system checks that the aircraft type assigned to a mission is suitable for the mission, and would warn the user if KC-135's were tasked for package force protection.

Taken together, this range of checks provides the ability for TEMPLAR to do extensive input checking. Adding constraints to the system, as discussed in section 2.2, would expand the range of checks performed. As delivered, the ADM does not include enough constraints to eliminate need for thorough review by skilled planners.

As implemented, the constraint mechanism does not distinguish between constraints which test conditions spanning multiple plan elements (e.g., missions) and those which have only local scope (e.g. the constraint on aircraft type above). The assumption that constraints all have global scope causes significant numbers of frames to be examined, and often creates a severe paging load on the system which detracts from performance. Because most constraints have local scope, adding the capability to specify that characteristic could speed operations considerably.

When a check is violated by a user input, TEMPLAR displays a popup window with text explaining the problem. For those checks based on the constraints, locality effects provided by the constraint mechanism allowed explanations to be more usefully generated from the log of failed and passed constraints for the value than from a complete computational history.

2.4 Flexible Man-Machine Interface

The TEMPLAR Man-Machine Interface style of interaction has proved to be extremely effective in support of the user and his ability to get the planning job done effectively. The fact that the user can freely switch between the various forms of interaction, between and even in the middle of commands, demonstrates the flexibility which can be provided in a C^3I system. Only in rare circumstances is the user locked into a two- or three-keystroke sequence of command processing when such switching is disallowed, and even then the user can abort (and thus bypass) the completion of the command sequence if desired.

2.4.1 Natural Language

In general, a natural language input interface is not appropriate in situations where the users know their job because of its extremely low bandwidth. It just takes too long to type English sentences (whether syntactically correct or somewhat abbreviated) compared to how long it takes to type single-word commands, do single key-stroke commands or move a cursor via a mouse. Natural language input interfaces are appropriate where naive users without knowledge of the problem area can ask in their own terms about areas of the system, e..ner for general browsing or during training in the problem domain and the use of the system.

This picture is further exacerbated by the amount of computation required to handle both a broad subset of English syntax and common language, and a large problem domain which is rich in its internal terminological refereats. Natural language grammars of this quality must be large and parsers are correspondingly overburdened. The TEMPLAR natural language input interface has a mean response time of 30 seconds to effectively parse a sentence. The coverage provided by the five grammars is large, but then so are the grammars, themselves. In the final analysis, the parser simply does not have the horsepower to process grammars of the size and complexity present in TEMPLAR at any reasonable speeds -- something which was not foreseen in sufficient detail until the fifth grammar was nearly completed (although there were early concerns about the problem which resulted in the grammar structure being partitioned into the five modules).

The TEMPLAR users have the expertise to perform their jobs efficiently and hence a natural language input interface to the system is unlikely to be used even were it to provide an instantaneous response time. As a result, as the project progressed, less time was devoted to polishing this interface and more time was devoted to supporting other aspects of the Air Tasking Order planning problem. Consequently, while the decision tree mechanism used to dispatch the indicated commands based on the parsed natural language input was very effective and efficient, few TEMPLAR commands were made accessible. The decision tree was not fully populated, but treated rather as a proof of concept in the final linkage in the natural language input interface to controlling the system. Also, the natural language grammars and parser included both the ability to mix English words and mouse pointing input and the ability to detect anaphoric references to objects mentioned in earlier natural language input based on the type of that input. However, while these features are present, they are not currently hooked up to the decision tree mechanism.

The English explanation of the Package Planner's reasoning was output in the Combat Plans terminology and was very effective, but it was also voluminous. An autoplanning run of a full-up Air Tasking Order will typically produce 250 pages of detailed explanation. This volume of output was cause for concern, but firstly it would be difficult to cut down in size without risk of inadvertently misdirecting the user. Secondly it was felt that the user would typically never analyze the bulk of the data, but rather pick and choose the details of importance. The output of the explanation did serve to slow the Planner somewhat, but its performance was adequate to meet the necessary criteria. A mechanism of duplicating the explanation in a file for later perusal was implemented, but the invocation switch was not provided at the user interface level (i.e., in the system as delivered it is turned off, but can be turned on by a programmer). It was unclear whether this would be of aid to the users.

The English explanation of constraint violations and enumeration orderings was effective. However, few constraints and preferences were implemented in the non-inverted version which allows them to participate in explanations. Both explanatory capabilities used the same mechanism -- a straight-forward approach of using predeveloped textual messages filtered by the context of the current situation. Where constraints and preferences were implemented as inversions in order to bring to bea the much better execution performance of the inverted style, explanation was unavailable due to the nature of the preference inversion mechanism. It would be reasonable to provide a non-inverted version for every inverted constraint or preference, and to support both explanations and enhanced performance for each such pair of versions; however, while this was initially investigated, it was not pursued.

The English explanation provided by the help mechanism for the active field of a form was adequate, but the examples provided were not tied to the enumeration mechanism. Had they been, the examples would have been more effective, and more closely reflect legal choices given that they take into account the context of the situation.

2.4.2 Menus

The TEMPLAR system of menus allowed the user to control much of the TEMPLAR functionality without overburdening the user with a massive selection at each step of the way. This was accomplished through several different hierarchies of menu functionality. The fixed form action menu (lower right of each form) and variable form navigation menu (upper right of each form) were effective in bringing the right set of commands to the screen at the right time. The TEMPLAR system of menus provides a particularly good example of menus whose list of selectable options/commands is tied to the current context, thus in large measure putting the appropriate functionality at the right place and time.

The other hierarchies were typically two-level mechanisms where the top level would serve to select major function classes and the second level would serve to select the particular function of the class.

However, given the number of menus available within TEMPLAR, the branching factor and number of levels in the hierarchy, in many instances even experienced users occasionally had difficulty determining where a particular remembered menu selection was within the hierarchy. A fast solution to this, implemented for the forms navigation hierarchy, was to provide two versions of a flat all-forms-shown menu. Each was, and needed to be, scrollable due to the number of forms available in TEMPLAR. A second solution which was analyzed and considered feasible, but was not implemented, was a user-defined macro facility which would have allowed the user to associate a specific keystroke to the selection of a specific form from anywhere within the navigation hierarchy.

The map pull-down menus at the top of the color screen have proved too slow in their response time. This is due to the factor of eight increase in the number of bit-planes which needed to be saved :.1 order to preserve the map areas under the pull-down menus. To partially offset this, many of the menus and other informational pop-up messages were displayed on the B&W screen even when the user's focus was on the color screen. In retrospect, it is likely that all such temporary displays should be moved to the B&W screen -- the increase in response time to reasonable levels more than offsets the momentary distraction of the requiring the user to change his focus from one screen to the other.

2.4.3 Graphics

TEMPLAR provides a very effective map graphics display system. The software overlay approach was very flexible in economically allowing the selective erasure of individual screen items. The use of object-based programming mixed with frame-based programming was extremely successful. We used the Symbolics' native Flavors object system rather than Knowledge Craft's CRL object mechanism due to unusually poor execution-time for the latter as an object programming support platform.

The fly-leaf corner maps allowed cut and paste operations and the ability both to view multiple disjoint geographic regions and to view the same region at differing scales and with differing iconic overlays. The icon highlighting via the mouse coupled with the simultaneous display of the icon's geo-object name on the status line allowed for quick identification of objects in a cluttered cluster. The duplication of the B&W mouse-button command line prompts on the color screen as well served to reduce disrupting change of focus and consequent user distraction. The straight-forward point-grab-move map object modification mechanism was simple and easy to use.

On the down side, the software implementation of map pan/zoom functionality was too slow. To counter this, three different map display algorithms were implemented in trying to remove the response-time sluggishness, but none of them was wholly satisfactory. The final response time for a map zoom was approximately 30 seconds, but we feel that a reasonable response-time should be on the order of 3 seconds or less. Unzooming was sped up by caching the prior screen contents on a stack prior to a pan or zoom, so an unzoom required only about 5 seconds. Only the map backgrounds were zoomed, the icons themselves were overlaid on the background following its pan or zoom.

The algorithms involved were:

1) an initial simple pixel replication which provided a response-time in the neighborhood of twelve minutes per zoom.

2) a digital differential filter (DDA) adapted to two-dimensional scaling which produced a fairly uniform 30 seconds per zoom.

3) a quad-tree implementation which was significantly slower at low magnifications but became faster at replications of x16 and higher.

The quad-tree approach had two drawbacks. The first was that in order to provide reasonable map background detail we had provided two scales of map (the mechanism is built to handle more) and switched from the low resolution map to the high resolution version when the magnification got high enough. This made the quad-tree less effective until the higher resolution map was finally magnified to x16 and over, an infrequent occurrence. The second problem was that the quad-tree approach required many more megabytes of storage, which further slowed system performance due to paging and disk storage requirements for the extra virtual memory involved. We did implement a hybrid approach where the DDA was used on low magnifications and we switched to the quad-tree above a given threshold, but the second drawback to the quad-tree approach effectively removed it from competition.

We could expect further improvements in the DDA if map pages were frozen in physical memory before the DDA algorithm began its replication, but this was not implemented. Unrolling the innermost DDA loops might also buy some time, but this was not investigated. The most effective approach would be specialized map hardware-assist for both the object iconic overlays and the pan/zoom functionality. We decided that we could not use the built-in color hardware pan/zoom due to 1) it is unavailable on the CAD color hardware which was required for its flicker-free display, and 2) it replicates based on the entire screen rather than by window. Additionally we would have to erase the icons at outdated screen-positions as well as repaint them, plus we would have to erase as well as repaint the other informational and map windows on the color screen. The DDA algorithm also performs scale reduction as well as magnification, which is not supported by the hardware. In fact, for one of our map backgrounds, the initial display is a reduced-scale version of the low resolution map and the map's full details only show up after the first zoom.

Other minor issues also presented themselves after user-testing. We were limited as to the colors we could display by the 8-bit CAD display limitation -- 24-bit color would allow a wider variation in displayable objects and color highlighting. We mention an 8-bit limitation rather than include the extra 9th CAD bit-plane both because it is accessed via a slightly different mechanism, and because we had decided to implement graphics software which would run on any of the Symbolics' color screens. The fonts used on the color screen were a shade too small. A slightly larger more-readable font size could be employed, and displayed in reverse video to make switching vision from the B&W to color text easier to follow. Moving the mouse between the B&W and color screens would have been more convenient with an even simpler mechanism than the Symbolics' built-in two-keystroke command.

2.4.4 MMI Context

A variety of mechanisms were implemented in TEMPLAR to support context sensitivity during MMI input. Each field of each form was tagged to know about the type of [illegible] expected for that field, so that both the "next character" typed into the field and the final completed text typed into the field were separately validated to provide the earliest possible feedback to the user on invalid entries to the fields. In addition, the final field's value was also processed through a semantic validation based on the current state of the knowledge base. These validation efforts are in addition to any applicable constraints which might influence the validity of the field's value.

Contextual sensitivity based on mouse focus is exemplified in several ways within TEMPLAR. Both the map object icons and the form fields exhibited highlighting when the mouse cursor was positioned over them. This reflected a mechanism to dynamically alter the currently available sets of commands based on what the mouse cursor was over. A second type of highlighting was displayed when a specific field had been made the focus of activity, and here again yet more commands based on the particular field became available. Automatic display of a map object's name based on the currently highlighted map icon has already been mentioned above. In addition to this, the geographic location is continuously tracked and displayed for the user while the mouse cursor is over an exposed map.

While the user is working on the maps, the mouse-button status line provides prompting help based on the stages of the current multi-keystroke command in progress. Menus whose list of selectable options/commands is tied to the current context of operation proved to be effective and efficient. Some of these mechanisms were based on standard Symbolics' operating system mechanisms; however, most represent an additional layer of design and implementation.

The ability to selectively inhibit entire classes of map objects as well as inhibit individual map objects from appearing on a map display went a long way toward providing the decluttering required for users to be able to easily perform their tasks. While user job models provided an effective way to inhibit users from accidentally modifying data outside their scope, it now appears that extending the two decluttering mechanisms to provide standard retrievable map tailorings tied to job models and, indeed, to individual users, would serve the user even more effectively in decluttering the maps and focusing attention on the task at hand.

2.5 Knowledge Representation Modifications

TEMPLAR provides the ability to modify the knowledge base directly through the Knowledge Craft Palm editing facility. However, we deem the editing of the knowledge base structure extremely dangerous at the user level due to the ability to inadvertently and significantly detune the system, or to destroy key logical invariants upon which the system depends (i.e., to destroy the logical consistency of the knowledge base). *Effective manipulation of the knowledge base structure requires both Lisp and AI programming experience and an intimate understanding of the relationship between the knowledge base and the functionality which manipulates it.*

Certain types of objects, map objects for example, are modified by the user in a reasonable way through the Knowledge Craft Coconut editing facility.

The TEMPLAR Formal Constraint Language provides the bulk of the material needed for on-line Natural Language constraint modification, but the user interface necessary to directly allow the users to change constraints was not hooked up. Again, such user modification is considered extremely dangerous due to the ability to inadvertently detune the system. *It would be moderately straight-forward to provide a TEMPLAR run-time constraint definition facility through a form, but it is unclear whether domain knowledgeable users would benefit from such a capability.*

2.6 Mission Task Planning

TEMPLAR is able to plan all the mission types specified in the Statement of Work. Four distinct planning approaches, discussed in section 2.1, were used to achieve this capability. The need for a variety of planning mechanisms, a requirement not found in KNOBS, is a direct result of the expansion to planning multiple mission types and planning multiple plan elements simultaneously. The need for the package planner to plan mission lines on the Target Planning Worksheet is established. The package planner is most closely derived from the technology in KNOBS, but incorporates a number of additional elements to handle the problems imposed by needing to plan multiple elements. With what we know now, the package planner could be stratified to permit a comprehensive integration of least-commitment planning techniques with the depth-first walk. As is, the MMI mechanisms in TEMPLAR can support user interaction with the abstract planning strata needed by such a planner.

The requirements for the other three planners are explained below:

- o The flow planner derives from the fact that CAS and DCA missions are not scheduled to achieve a specific result over a specific target, but rather to provide a level of airborne support in an area over a range of times. Repetitive depth-first tree searches could have been used to schedule these missions, but were not required. Instead, a cyclic algorithmic process applies a relatively small number of constraints on distances and availability in order to achieve the necessary aircraft flow.

The flow planner operates from flow specification forms, which provide a set of lines tr input the priority, station duration, mission interval, number of aircraft, etc. The planner processes lines independently in the order determined first by stated priority, if any, and second by order on the form. For each specification line, the planner fills in a set of slots (unit, aircraft type, ETOT, ETFT, and number of aircraft under all circumstances; SCL, comments, and alert code if required).

The planner bases its processing on the sortie flow allocation capabilities internal to TEMPLAR. The unit and aircraft type chosen by the flow planner are that which are in range of the area using the given SCL (if any) and which have sufficient available sorties at the proper time and which are closest to the target area. The ETOT/ETFT are computed directly from the values on the specification line. The SCL, if any, is taken from the input on the specification line, and if supplied is used to constrain the selection of unit and aircraft type to those which carry the given SCL. Because the number of combinations of unit and aircraft type which need to be tried is very small, the flow planner can perform an exhaustive search once the legitimate candidates have been identified.

o The air-air refueling planner performs a great deal of algorithmic processing, followed by a loop similar to that of the flow planner to finally allocate aircraft to fuel requirements. The sub-elements of the AAR planner are as follows:

1. Determine ATO-wide fuel requirements.
2. Group pre-attack tanker cells with their packages and form tanker missions for those cells.
3. Compute a reachability graph detailing which fuel requirements can be reached by a tanker flying from prior refuelings.
4. Map tanker missions over the reachability graph to form a spanning set of missions.
5. Allocate aircraft to tanker missions.

Despite the fact that it appears to make sensible planning decisions, the AAR planner is, essentially, a first algorithmic implementation of an ill-defined problem. It needs tuning to refine some of the planning estimates it incorporates, and needs validation of some of the heuristics incorporated into its graph algorithm. It also needs more accurate fuel flow models than were available for implementation in TEMPLAR.

In addition, the size and processing requirements of the AAR planner are such that, taken by itself, it could be implemented on a 386-class portable PC. This would provide a powerful tool for replanning on-board the AWACS.

o The PCC Planner fills in the mission number, call sign, SIF code, and mission role (e.g., AR, DCA, etc.) information into mission lines. The planner is constrained to compose these values according to guidelines published in the Knowledge Book part of the TEMPLAR Maintenance Manual; the implementation of those guidelines is implicit in the code for the planner. Given the ability to compose valid field values, the planner uses a simple allocation process to determine which of the valid compositions are available for the necessary time interval, if any. If none of the compositions are available, new elements are drawn from an available pool and used to compose new value resources.

For example, call signs are stored by call word, where a given call word is assigned to and tracked on the basis of a subunit. For each call word at the subunit, the times at which each call sign composed from that call word is available are recorded. When a new block of call signs is required for missions of that subunit, the set of existing availability intervals is queried. If a suitable block is identified it is used, else a new block is opened.

2.7 Knowledge Base

The TEMPLAR development clearly demonstrates that frame-based programming works well in dealing with large-scale models of the world. The multiple interpenetrating frame hierarchies[6] provided more than adequate modularity to control the growth of the world model. The frame-based approach provided considerable flexibility during development in the ability to alter the world model to take advantage of new information about the Air Tasking Order problem domain as it became available.

The TEMPLAR knowledge base is extremely large, consisting of 50,000 to 100,000 frames. The modelling achieved by the TEMPLAR knowledge base is, in detail and in interaction, several orders of magnitude beyond that which can be handled by the current state of the art of rule-based systems. In spite of this, transient frames are rarely built. In retrospect, now that the bulk of the domain knowledge has been acquired and digested, it looks as if the knowledge base in terms of frames instantiated and hierarchies dynamically

[6] That is, from one perspective the directed graph which is the knowledge base consists, from one perspective, of multiple trees rooted at various key classes (e.g., mission, target-mission-group, map-object, etc.). Relations form the arcs of these trees. Because one frame will often be a member of more than one tree, the text refers to these connected trees as "multiple interpenetrating hierarchies".

modified is nearly constant. There has been serious consideration given to an AI development methodology in which the near-static frame-based world model of a problem domain such as TEMPLAR is initially developed in Lisp and later converted to an equivalent knowledge base representation in a conventional language. Such a methodology seems feasible as long as the final knowledge base is, indeed, fairly static in nature.

The layering of TEMPLAR-internal objects, static domain knowledge, long-term slow-changing domain knowledge, and daily-changing domain knowledge was very effective as an additional approach to controlling knowledge base modularity and growth. The earliest laid parts are still rough. Due to its nearly continuous growth over several years and due to several course corrections which took place during that growth, there are different areas of the knowledge base which achieve the same effect through different mechanisms and styles.

An early and inadequate representation for time led to a later upgrade of the representation, but this generated several deep-seated bugs in the temporal reasoning which were only lately discovered[7]. The time representation was also a bottleneck in speed of execution of many aspects of the resource accounting performed by TEMPLAR. Special attention could be paid to designing for faster execution rather than simplicity now that a working version exists.

The TEMPLAR system takes a bipartite view of the knowledge base. There is the world model itself and then there is the parallel knowledge base which maintains knowledge about how to display geographical objects on the map. Not all of the objects modelled within TEMPLAR are map displayable, of course. Those which are each have a dual map representation frame linked into its display hierarchy. This separation of the world model from the information required to display it on a map has proved to be among the most successful partitioning in the system. A similar partition with respect to the objects displayed on the B&W forms turns out not to have been needed due to the fact that all fields which are "windows" onto the knowledge base are displayed alike. The key, here, was the multiplicity of differing display techniques which were required for the map iconic display.

TEMPLAR uses frames to model its own internal computational objects, such as the

[7] In particular, the initial representation of time did not adequately distinguish time intervals from absolute times, and did not adequately support the representation of ATO periods which included midnight within the period. When the changes were made to resolve these problems, a number of difficult problems were uncovered which were due to programmers having referred to elements of the time data structures directly rather than through access macros. Because the standard access macros had not been used, the code did not properly respond to the representation change.

B&W screen forms, as well as real-world objects such as aircraft and airbases. The frame-based approach allowed us to easily add meta-knowledge, a layer of introspective knowledge about the TEMPLAR processing itself, to control the system. Such reasoning was used, for example, to ensure knowledge base consistency and integrity across workstations in order to allow multiple users to be planning aspects of the same Air Tasking Order simultaneously.

The TEMPLAR knowledge base used a wide variety of frame inheritance types. In rare circumstances when the inheritance was performed many times and included computationally expensive deductions, the addition of a caching/invalidation mechanism served to provide the performance needed.

The Knowledge Craft CRL context mechanism proved to be inadequate in controlling the fine detail contextual merging required for TEMPLAR. Hand-tailored mechanisms were built in its stead.

Also, while using Knowledge Craft CRL as an off-the-shelf frame language platform initially saved some development time, the lack of adequate software support for CRL in identifying and fixing bugs and in lifting useful features from within CRL up to the CRL-user level, coupled with our inability to obtain a source code licensing agreement for CRL, forced us to both bypass CRL mechanisms in some instances, and to wire changes around other CRL mechanisms to prevent unwanted behavior.

2.8 Planning

System-wide, TEMPLAR can meet the performance requirements in the SOW, although performance of the planner clearly depends on how well the weaponeering matches the available resources. Pathological cases can undoubtedly be constructed which severely extend the time to plan. The performance requirements are poorly defined, however, and it is not clear how well they correspond to the needs of the operational user.

- It is unlikely that operational users will request the system to plan the entire ATO. Accordingly, system performance in terms of planning speed and quality for specific planning tasks is much more significant than a measure of overall ATO planning speed.
- The user/system interaction performance is also important, since it is a combination of user and system actions that will produce a combined ATO. Measures of the combined user/TEMPLAR system should be developed and evaluated.

TEMPLAR requires approximately 200MB to store the ready-to-run image for the Blue Flag or CalNeva scenarios. Of the storage required by internal data structures, we estimate that possibly as much as two thirds of this volume is required for storing meta-slots in the Knowledge Craft CRL representation. With the perspective of having completed the software, it is clear that the representation provided by CRL is much more general than is required for TEMPLAR, and that a replacement for CRL sufficient for TEMPLAR purposes could be written with the following characteristics:

- Be interface compatible, not requiring changes to the TEMPLAR software (with the exception that the Natural Language interface, which relies on much more of CRL than does TEMPLAR, would be removed).
- Be smaller and faster than CRL, possibly by a factor of two. This might be sufficient to permit full-scale operation of the software on a Symbolics 3650 having only one 368MB disk.
- Provide full source code and data rights to the Government, enhancing maintainability of TEMPLAR.

2.8.1 Autoplanning

We found that the KNOBS style of automated mission planning based on filling in the slots of a single frame was adequate to the smaller scale problem on which it was used but required a meta level of processing to deal with the Air Tasking Order problem domain. The number and quality of interdependencies among the various packaging problems, the criteria in determining an adequate choice in planning an ATO, are severe. Some of the interdependencies taken into account within TEMPLAR involve the following issues, for example:

- between the attributes of a mission: unit, airframe, preferred unit role, airframe role change, ordnance load, drag
- between the missions for a target: attack, attack designation, force protection, defense suppression, electronic combat, reconnaissance
- between targets which are packaged together: target co-location, enroute threats, air-to-air refueling locations, offload rate and quantity, aircraft-boom ratio, whether to pursue economies of scale[8]

The level of package planning expertise within TEMPLAR is difficult to judge. Some aspects of the planning process as implemented in TEMPLAR are fairly naive, while others are quite elaborate. Additionally, as with many automated systems, the raw cross-

[8] The decision of whether to package a target with another target, and if so with which target, is particularly difficult. Economies of scale occur when support missions can be shared among the attack missions for multiple targets, but the decision to package two targets together can result in many problems, not the least of which is insufficient tanker capacity.

correlative bookkeeping which TEMPLAR does accurately and quickly is quite beyond ordinary manual skills.

Complex, rare cases were not addressed. For example:

- no attempt is made to add tankers to the standard flow (this is left to the AAR graph-based planner)
- no attempt is made to deduce a need for a massive air effort (nor is the information to do so present), so no gorilla packages are created

Certain practices within TEMPLAR, which are performed apparently naively, actually reflect an obscure ordering in the current expert approach due to its present manual nature in handling the large amounts of information. For example, any package with enough unused offload and boom capacity will have a RECCE mission added to it whether it makes sense to send a RECCE mission in with the force package to the given target area or not. No further knowledge other than the RECCE mission will "fit" is used in its planning. The current expert practice is actually to lay in these missions blindly and use them as AAR place-holders to ensure that fuel and boom capacity are set aside. Later, the missions are reshuffled, when the general picture of the targets to be hit has settled down and can be manually cross-correlated to the reconnaissance requests of the day.

An early prototype of a least-commitment planning mechanism (present during the second, third and fourth incremental deliveries) which handled a mildly extended version of the KNOBS planning problem proved seriously deficient in terms of speed of execution. It is not active in the final TEMPLAR delivery. It is unclear whether such an approach, by itself, could prove successful in the full-up problem domain. However, as a quick rough-cut plan generator for identification of resource bottlenecks, we feel that a least-commitment planning approach can be used to advantage. Based on our experiences, both least-commitment and stratification should be considered in future package planners.

Numerous alterations were made to the final package planner due to knowledge which became available near the end of project development cycle. *While the planner is fairly effective, it is not clear whether the plan produced by the package planner is sufficiently robust under all, and especially pathological, planning situations*[9]. A serious impediment during the development life-cycle was lack of access in the early and middle stages of the

[9] Note, however, that this issues applies only to decisions made by the autoplanner. The mixed initiative planning capability permits any and all decisions to be made by the user, and permits those decisions to be made at any time. Consequently, cases which exceed the ability of the autoplanner are always subject to resolution by human planners; therefore when taken as a total capability, TEMPLAR does not have the limitation cited.

project to the Air Tasking Order planning experts. This adversely affected the depth of understanding of the ATO problem, and led to significant delays in the development of the important areas of the knowledge base and the various autoplanners. Late during the project some limited access was made available. However, while this access provided a wealth of detailed information, it also resulted in significant revamping of these areas of the system.

TEMPLAR was designed in such a way as to never overrule a user's decision. Related fields on forms were grouped into frames and each frame was tagged as to whether it contained a field which had a user- or autoplanner-supplied value. If such a frame was tagged as user-owned, only empty fields would be candidates for planning. As such, when autoplanning was invoked, the package planner would attempt to complete a partially filled in user plan by filling in fields around those the user had filled in, and by replacing values in fields previously filled in by the autoplanner. This feature is termed mixed-initiative. If some partially completed plan component was not able to be completed, TEMPLAR will go on to attempt to complete its own version of the component, leaving the partially complete version in the original user's state. In this way, TEMPLAR is able to bypass user-enforced inconsistencies in attempting to achieve a coherent plan.

The package planner organizes its development of attack packages into several phases. Overall, those phases are the following:

1. Calculation of estimates of aircraft usage assuming that each TPW receives its first weaponeering choice for its first DMPI.
2. For each TPW, calculation of the set of viable weaponeering/tanker area combinations (e.g., from which tanker areas, if any, can aircraft of the type specified in weaponeering line number 1 reach the tanker and the target from the bases at which they are stationed?).
3. For each TPW, calculation of the order in which the viable weaponeering options will be examined, assuming that the tanker area closest to the target is used.
4. For each TPW, each of which represent a nominated target, calculation of the other nominated targets which represent threats to aircraft enroute to the target at hand. This partial ordering is then combined with the target priorities, the day/night specification, and user-entered ETOT's, if any, to produce a total ordering on the set of targets to be planned. Each target also receives an interval which represents its earliest and latest permissible ETOT.

Once these preliminary decisions are made[10], the TPW's are then planned

[10] Technically, these four steps plus the final detailed planning would be the basis for a fully stratified planner

individually in the order determined in step 4, which may vary from the strict priority order to reflect threat dependencies. For each TPW, the following steps are carried out.

1. Attack missions are planned first on the TPW. The first (in the sense computed above) weaponeering option which meets all constraints is used. When tanker assets are assigned, the planner attempts to combine the attack on the "current" TPW with that on a TPW already planned for which the targets are sufficiently close and for which there is adequate fuel on the tanker to support the additional missions.
2. After the attack mission is planned, laser designator attackers are added if required by the attack weapons load.
3. Force protection missions are then assigned based on the degree of air-air threat expected. The missions scheduled must fall within per-package limits on estimated fuel offload from the pre-tanker, and can only be scheduled if sufficient sorties are available.
4. Wild Weasel missions are scheduled if terminal area threats are expected which can be defeated by such aircraft. These missions, and the EC and RECCE missions planned next, are also subject to availability and fuel constraints, and are not planned if those constraints are violated.
5. An electronic combat mission (EF-111) is scheduled if adequate Wild Weasel support was not available.
6. A reconnaissance mission is scheduled if fuel is available on the tanker and sorties are available. These missions are placeholders, intended to reserve fuel on the tankers (necessary since the recce target list is not expected to be available until somewhat after package planning is complete). Later in the planning process, these dummy recce missions are stripped out and the actual missions responsive to the recce target list entered.

To illustrate the idea of threat and ETOT dependency, suppose that an airbase is the priority 1A target, and a missile site immediately in front of it is the priority 1D target. Further, suppose that the missile site has been specified with an ETOT of 0730. Because the missile site threatens aircraft attacking the airbase, the missile site will be planned before the airbase, violating strict priority order. Further, the ETOT for the airbase will be restricted to be after 0730, the fixed ETOT for the missile.

It is a straight-forward procedure to perform an autoplan run, change some of the governing information (e.g., the weaponeering options for the attack mission), and reinvoke the planner again to see the differences in the resulting plan. In addition to the planning

using least-commitment planning approaches.

forms, there is a group of setup forms which provide initializing information about the static and slowly changing aspects of the war. Once planning has begun, these forms cannot be modified -- to prevent knowledge base inconsistencies. There is nothing unusual about these setup forms which prevents the user from being able to change the information thereon, but the required implementation to forward those changes to the already planned portions of the ATO is not present in TEMPLAR. Had the requirement to avoid the "replanning" problem not been levied on TEMPLAR, the required work could have been done. Replanning in the face of changes once the plan has been partially built is not a difficult problem, but it is a pervasive one. Parts of TEMPLAR handle what might be called replanning while other parts do not. The mechanisms needed for replanning support have all been proven to work. In the programmers opinion, the cascade of changes needed to handle replanning based on a change in some precondition is straight-forward to implement (albeit large-scale) and is implementable in a fundamentally tractable fashion in a high-performance planning system.

Of the four autoplanners implemented in TEMPLAR, only the package planner includes a generate-and-test component as can be seen from the planner explanations. In reality, very little is generated and tested due to the amount of knowledge from the problem domain incorporated into the planner itself. While extensive use of meta-knowledge is made during planning, no one simple generalized control mechanism or paradigm stood out as both sufficiently capable of handling the complex knowledge interrelationships and sufficiently fast to meet the performance criteria required. The Air Tasking Order planning domain has been categorized as of the "stereotypical planning problem" type. Nonetheless, there are significant aspects of other problem types present.

The knowledge gained during the development of TEMPLAR with respect to the issues of planner generality versus AI planning "paradigms" versus the "type" of planning problem under consideration has led us to the following conclusion. Specific planning "paradigms" have been associated with specific types of planning problems. We believe that for any sufficiently rich planning problems (and by this we mean problems which require both complex knowledge as opposed to complicated bookkeeping and which include bodies of knowledge in which people must be trained in order for them to perform well), these problems will all be seen as "stereotypical planning problems". Only the smaller simplified knowledge-poor problems can legitimately be viewed as of other problem types. Single paradigm approaches are seen, in this light, merely as higher-level weak-methods approaches in which the domain knowledge is forced willy-nilly into the chosen paradigmic mold. Our recommendation is to be eclectic, to fit the various paradigms to the problem and be ready to bend them as necessary -- let the knowledge be the guide.

2.8.2 Supply Tracking

TEMPLAR defines a hierarchy of weapons assemblies. Weapons themselves, which have no tactical unit association by type, are grouped into loads. A load is a set of weapons, with the quantity of each weapon in the load specified. A typical load might include a set of bombs plus the racks needed to mount them (note: racks, pylons, etc. are considered weapons in this context). Every load is associated with a tactical unit.

Loads are then grouped into SCL's. A typical air-air SCL might consist of two loads, one for AIM-7's and one for AIM-9's. Each SCL specifies the name and quantity of the loads it includes. SCL's are also associated with tactical units.

Mission ordnance is specified by filling in the SCL field for the mission. Supply tracking demons in TEMPLAR then process the hierarchy under that SCL to determine how many units of every expendable weapon in the SCL are being consumed by the mission. A report providing totals by weapon type is available as a form.

Forms are also available to enter availability information for weapons by airbase. Current weapons usage by airbase is also reported on those forms. The user is not informed when more than the available weapons at a base have been consumed during planning.

2.8.3 ATO Presentation

The ATO is visible in TEMPLAR via the extensive number of forms implemented. Most of the detailed mission data are available in the unit schedule forms, for which there is a command to display and print all non-empty unit schedules. Summary data are available in other forms. The data on the forms are kept current via a flexible forms display system, including the Forms Editor and Forms Compiler tools plus extensive MMI software to provide and update form-to-Knowledge Base links. That software also makes updates and changes to the forms straightforward. The tradeoff for that flexibility was that forms require extensive computation to build the first time they are displayed, leading to a setup requirement that forms should, to the maximum extent possible, be pre-built and saved to disk to enhance the speed of the system. As with most areas of TEMPLAR, tuning of the forms construction software for additional speed is possible.

2.9 Context Mechanism

The context, or "What If" mechanism in TEMPLAR is applicable to nearly all functions of the system rather than being restricted to developing force packaging options. Contexts (copies of frames which shadow the original versions) in the frame system provided the basis for this functionality. The context merge tools of CRL were completely inadequate, however, providing no user control, no user visibility, no ordering control, and no demon firing. In order to solve these problems, keep the knowledge base consistent with the network, and provide control/visibility, it was necessary to write software to perform context merge at the form field level.

2.10 Job Models

Job models in TEMPLAR are implemented using both a frame hierarchy and the LISP machine login process. The user name and specialized initialization actions (including setup of default printer access) is done via LISP machine login. Controls on forms accessibility, KB editor access, and other TEMPLAR-specific features are recorded in frames; the slot values are explicitly interrogated by the TEMPLAR software providing associated functions.

User tools to edit, create, and destroy job models would be useful, as would the addition of keystroke macros, display specifications, and map window setups.

3.0 TEST AND EVALUATION

3.1 Technical Evaluation Testing

The technical evaluation testing was performed as a part of the Phase III portion of the TEMPLAR contract. This testing was performed at TRW's AI laboratory, Systems Engineering and Development Division. The technical evaluation testing consisted of testing the capabilities of the TEMPLAR ADM against the technical goals as defined in section 4 of the "Test Plan (FINAL), TEMPLAR", CDRL A009, dated August 1, 1988.

3.2 Demonstration

As a result of contract changes only one demonstration was required. This demonstration took place at RADC the week of October 18-21, 1988. The demonstration consisted primarily of demonstrating capabilities not previously delivered or seen by the Government. The major capabilities demonstrated included the package planner, which replaced the previous prototype Least Commitment Planner, the flow planner, the air-to-air refueling planner, and the mission element planner. Also demonstrated were the job models, the CAFMS and LENSCE simulated interfaces, enumeration, ordering and constraint checking. Other capabilities demonstrated were portions of the technical goals as defined in section 4 of the "Test Plan (FINAL), TEMPLAR", CDRL A009, dated August 1, 1988.

3.3 Final Acceptance

The final acceptance test took place at RADC the week of October 18-21, 1988. The acceptance test followed the System Acceptance Test Description, section 7, of the "Test Plan (FINAL), TEMPLAR", CDRL A009, dated August 1, 1988. Since the demonstration preceded the acceptance test at RADC it was not necessary to rerun the demonstration.

4.0 SYSTEM DEVELOPMENT SUMMARY

4.1 Software Development Process

Much of the system was, after the fact, "conventional" software needed to automate the task in the first place. This includes software for user interface, ATO representation, system management, network control, etc. While ATO representation also includes Knowledge Base development, it is important to realize that all of this effort becomes much simpler if a conventional manual system already exists and can be used as a basis for development. To illustrate, suppose that a version of TEMPLAR providing all the forms and manual functionality but none of the planning assistance existed at the beginning of the project. By inference, then, the following work which had to be done for TEMPLAR, would have already been performed:

- The process and information flow for ATO planning would be defined in complete detail. This importance of having this cannot be overemphasized. Although the conventional system would, in essence, amount to a relational data base with forms for completing tables in that data base, knowledge of those tables, relations, and the flow of information from one to the next is a major requirement for Knowledge Base design.
- One version of a user interface would have been defined. Ideally, user comments on its strong and weak points would be known.

The development of TEMPLAR was such that the "AI" components were developed in parallel to these conventional elements. While it may be the case that this reduced the calendar schedule, it is clearly the case that it lead to a great deal of rework when prematurely-developed ideas did not correspond to the requirements of conventional ATO planning. An earlier emphasis on clearly defining the conventional system processing would have solved many of these problems.

If TEMPLAR were to be re-designed and re-coded, much of the conventional code could be written in conventional ways, with conventional languages and ported to conventional hardware. It is our belief, however, that there are core elements of the planner for which adaptation to conventional languages and hardware would be a gross mistake. Instead, we recommend that future programs embedding AI into operational systems use a hybrid approach, embedding AI hardware (such as a co-processor board) into conventional workstation platforms, and achieving a combined software suite where conventional functions are implemented on the conventional side and AI functions are implemented on the LISP side.

It is an incorrect extrapolation, however, to assume that implementation of the conventional software elements, much less the AI elements, should be done conventionally, as these statistics show:

	Lines	%
Operational LISP Code	185,787	55.24
Operational C Code	8,892	2.64
Forms Editor	11,437	3.40
Forms Mapping Data	11,314	3.36
Forms Compiler	5,414	1.60
LISP Source Code Control System	4,860	1.44
Utilities and Tools	1,522	0.45
Classified Scenario Data	26,232	7.80
Total Hand-Coded Software	*255,458*	*75.96*
Form Definition Files	2,624	0.78
Window Definition Files	78,206	23.25
Total Machine-Coded Software	*80,830*	*24.03*
Total TEMPLAR Source Code	336,288	

Other indicators also support the notion that the TEMPLAR development process was fundamentally different from conventional development:

- There are no source code listings of the system. Programmers used the on-line source code, assisted by the tools available on the Symbolics workstations.
- Despite the size of the system, configuration control, integration, and system builds required less than a one-person level of effort. This too was a result of the tools inherent in the Symbolics LISP machine plus the simplicity of building additional software tools in that environment.
- The actual lines of code developed were significantly higher, since a number of major elements were re-written when new knowledge came to light or when initial implementations proved unsuitable. Some elements were re-written several times. Including re-work done in mid-stream to adapt to a major operating system change, it is possible that as much as 60 percent of the software was re-written.
- Capital investment per programmer was significantly higher than on conventional projects in which multiple programmers share a single conventional workstation, or in which multiple programmers share a single mainframe via terminals. In the case of TEMPLAR, the average workstation cost was approximately $100K, and on average every programmer had a dedicated workstation. When network code was being developed or tested, a single programmer could productively occupy 2-3 workstations (this was also true when new versions of the system were being built). Even including the capital costs in the cost per line of code and backing

out extra hours worked by the programming staff, TEMPLAR was still remarkably productive.

4.2 Random Observations

A number of other issues arose in the development of TEMPLAR beyond those discussed earlier in this report. The following attempts to identify the more important of those issues and suggest how future projects can deal with them advantageously.

4.2.1 AI Tools

The use of purchased tools, such as Knowledge Craft, enabled a faster start after coding began, but ultimately hindered development. In particular, because of the central role of the frame system in the design of the software and because of the complexity of the TEMPLAR Knowledge Base, the inability of TEMPLAR to obtain an agreement from the vendor for source code access to Knowledge Craft slowed development materially while workarounds for problems were sought. The generality present in Knowledge Craft, not all of which is required for TEMPLAR, ultimately made the delivered ADM larger and slower than needs be. We estimate that a new frame system, replacing the elements of Knowledge Craft used by TEMPLAR with a TEMPLAR-specific design, can be written straightforwardly so as to both increase speed and decrease space requirements by a factor of two.

4.2.2 Communication Interfaces

Communications interfaces are both critical to operational use of TEMPLAR and were, as is for most operational systems, a major source of problems. In order to have the information base for planning, TEMPLAR requires input of large quantities of data. Much of this data is held in the LENSCE system, yet a lack of suitable communications interfaces (and definitions thereof) made it impossible to construct the electronic link. The LENSCE input link finally implemented was to have been tape-based; delays in receipt of test data made interoperability testing of even the tape interface impossible. Lack of interface definition caused several key information elements to not be available for extraction from the interface data records, even though they are apparently there somewhere.

Similarly, information barriers made it impossible to test TEMPLAR for interface with the CAFMS system and (worse yet), made it necessary to use a clumsy approach of terminal emulation via personal computer to transmit the data. This is so despite the fact that the new version of CAFMS is hosted on a VAX, and can apparently be run on the same physical VAX as was delivered for TEMPLAR. Logically, the interface between the two systems should have been via a shared database. (Actually, the same can be said of LENSCE; again, what was delivered falls far short of the integrated, interoperable software suite on a single VAX host that should have been possible.)

The operational consequence of not having been able to overcome these communications interface problems -- management problems and not technical ones -- is that the data which should flow into and out of TEMPLAR smoothly and rapidly is bottlenecked, and may end up being hand typed. This defeats much of the advantage of automation in the first place.

4.2.3 Map Backgrounds

Because so much of ATO planning is involved with maps, the map displays and map backgrounds in particular were critically important. Initial TEMPLAR maps used World Data Bank II country outlines; the proposed map backgrounds initially were to be derived from DMA digital data.

Unfortunately, comprehensive DMA digital coverage in the area of interest was not available, and what data did exist was difficult to obtain in large volumes when requested. As a result, the program approach was changed to use map backgrounds derived from DMA video disk maps. Although in general this approach worked well, the process and results had the following problems:

- The map data base is large no matter how it is stored, and when processed via the CPU and bitmap graphics in a workstation, is slow to access from disk and to manipulate.
- The video images are in NTSC video format. Hundreds of video frames had to be spliced together to form a single composite bitmap of the background. In the process, the color maps of the images had to be standardized (and constrained to a smaller set of colors at that), and the offsets and overlaps between frames resolved. The video disk images were not constructed with this in mind. Severe keystoning from the top of the map to the bottom occurred, as did major color shifts from one map zone to another. The resulting process we used was expensive and time consuming. Automated aids to the splicing process were not possible, as the format of the data files showing the geocoordinates and video disk address of a given frame is proprietary to the developing company and was not available to the project.
- In order to provide the maximum color fidelity, background color maps were derived individually for each map background processed. Because the Symbolics (and most other workstations) color hardware only supports color maps on a per-screen basis, it is not possible in the current TEMPLAR software to support the use of multiple bitmaps providing varying degrees of resolution (e.g., 1 to 2 million as well as 1 to 14 million). Constraining both images to the same color map would permit simultaneous display of both resolution images, but would degrade color fidelity. Color hardware supporting additional planes would also solve the problems, but would increase the hardware cost and increase the size of

the saved images.

4.2.4 Access to Experts

In retrospect it is clear that more access to experts was needed. *We also needed a complete planning walkthrough from those experts at the beginning, yet didn't really have one until the very end.* Funnelling dialog with experts through some intermediary is a sure way to kill a project such as this. We were fortunate to be able to talk to the experts directly at times. Best would have been to have had resident experts for the first 6 months and last 9 months, plus telephone calls and meetings in between. The degree of user involvement needed was underestimated by us and by the Air Force.

4.2.5 Implementation Language and Hardware

Parts of this system will be difficult or awkward to convert into conventional code, and the result will be hard to maintain. This is not to say it cannot be done, but to say that the decision to implement an ATO planning system in C on strictly conventional hardware is not one to be made lightly. The major issue is memory management; others include the widespread use of eval and apply. The resulting Ada or C software will be difficult to expand and enhance.

For example, the network implementation relies on passing small code fragments to other machines for execution. This is a powerful, flexible mechanism that supports a wide variety of operations without modification. While the networking in TEMPLAR can certainly be done via conventional approaches that rely on messages and code to interpret those messages, it will be larger and more brittle than the current LISP-unique approach.

As another example, it is commonly held that LISP (and LISP garbage collection in particular) is incompatible with operational systems. While garbage collection is certainly a concern in the design of such systems, it is worth observing the following facts:

- The design of TEMPLAR is such that users will re-boot the system once per day, starting over each day with a fresh system.
- Most of the data structures in TEMPLAR are either permanent and pre-built, or else highly transient and therefore quickly and efficiently reclaimed by the Symbolics ephemeral garbage collector.
- On a machine with approximately 726KB of disk attached, so that a large swap space is available, the onerous, hours-long garbage collections associated with the LISP stereotype are not required during normal TEMPLAR operation.

The argument that the Air Force cannot maintain LISP is not a sound reason for wholesale conversion to conventional hardware and C. Despite LISP's relative obscurity among the ranks of common languages, that will not be the issue making organic support of operational AI systems difficult. Regardless of language, the software which achieves the broad, powerful functionality of TEMPLAR (and perhaps any operational AI system) can be very large and very complex. Maintenance of such software requires very highly skilled programmers familiar with the AI technology used in the system. Examples of these technology elements include frame systems, inheritance and object-oriented programming; none of which are unique to LISP. If anything, LISP simplifies the maintenance problem in the sense that it removes the necessity for the programmer to be concerned with housekeeping details which will surface in a C or Ada implementation.

The critical implication of this is that existing Air Force software support assets must change to meet the challenge of operational AI systems, and that the nature of the plan to do that support will have greater impact on the choices of implementation language and platform than any other factor. Without such a plan, decisions about the supportability of one or another implementation language and platform are difficult to justify as more than informed opinion.

4.2.6 The Missing Tasks

As an Advanced Development Model TEMPLAR did not answer the following operational related questions:

- o What are appropriate methods and techniques for validating the knowledge and methods incorporated into an operational AI-based system?
- o What are appropriate methods and techniques for collecting and inserting new or changed methods and procedures into an operational AI-based system?
- o What are appropriate methods and procedures for providing software and hardware support for an operational AI-based system?
- o What are appropriate questions, methods, and procedures for evaluating the performance of an operational AI-based system?
- o What are appropriate test methods and standards for turnover of an operational AI-based system to the user community?
- o How can the results of this evaluation be incorporated into future system development efforts?

The answers to these questions are critical for future planning of programs which intend to use AI in an operational role. Because of the differences between laboratory evaluations and operational use of a system, it is simply not possible to answer these questions with a high degree of confidence other than through development and operational test and evaluation.

4.2.7 The Cost of Incremental Prototypes

From a development standpoint, agreeing to deliver the internally-scheduled developmental prototypes was a serious mistake. These prototypes, originally scheduled as internal systems-integration milestones at approximately 4-month intervals, became formal contract deliverables as part of a contract modification.

Unfortunately, we seriously underestimated the effort required to transform these internal prototypes (which as planned were not completely debugged, and were really only suitable for TEMPLAR programmer use) into ones which could be released for examination and evaluation by non-programmers. A great deal of re-work and lost schedule was caused by this decision. The problems encountered included the following:

- Mechanisms which were incompletely built could not be left incomplete, as users might stumble into them and hence into the debugger. Accordingly, either a throw-away mechanism had to be built in the interim, wasting effort, or else the real mechanism had to be built quickly, without sufficient time to think through all the implications. Examples of TEMPLAR software which were thrown out and completely re-written due to this effect include the initial versions of the Guidance and Allocation forms, which were too slow to be used, and the map edit software, for which there was insufficient time in the initial release to connect it to the Knowledge Base. There were many others.
- Since Integration and Test (plus writing the User's Guide sections and editing that document) consumed 3-4 weeks prior to delivery, and since preparation, conduct, and wrapup of Knowledge Acquisition sessions consumed 2-3 weeks after each delivery, only 6 or 7 weeks of design and development (i.e., about 50% of the time) were available every 13 weeks.

This is not to say that the idea of incremental deliveries from the project was in itself wrong -- the major problem was that the deliveries were scheduled too frequently, and there was no flexibility in their dates to allow for solving development problems.

4.2.8 On Appropriate Technology

It has been observed that TEMPLAR uses 10 year old technology and therefore is in some senses a technology failure. This criticism is correct in fact but irrelevant in conclusion. The useful question to ask, rather than whether or not the software incorporates the latest in AI technology, is whether or not TEMPLAR applies appropriate technology to the problem being solved and whether or not it applies it effectively.

We believe that the frame technology used in the TEMPLAR knowledge base is the correct technology for use in representing the force planning problem. TEMPLAR has added to this technology in at least these ways:

- TEMPLAR showed that a flexible, responsive, powerful user interface can be merged into a frame system, and indeed can apply the power of that representation to the problem of simplifying a very complex user interface.
- TEMPLAR showed that the scale of the operational force planning problem is such that rule-based systems are impractical for its solution. The TEMPLAR Knowledge Base consists of about 70-100 thousand frames once the system has been fully initialized. This is significantly larger than most knowledge-based systems, with strong implications for the sizing of the hardware platform and for selection of the representation and inference-generation technology.
- TEMPLAR showed that rule systems are desirable as integrated components of frame systems, however, and that existing, state-of-the-art knowledge base products do not provide the necessary integration of rule and frame systems. One possible formalization of the techniques applied in TEMPLAR would be to apply RETE net-type rule systems in which the set of active rules and the current working memory content is determined by the system's focus and inheritance from the frame system. This is a significant generalization from the single-rule-set/single-working-memory capabilities of current rule-based technology.

4.2.9 Network Considerations

TRW proposed a networked system of TEMPLAR workstations in order to be responsive to the operational requirement of involving many individuals in the planning process. This requirement has been validated through the TEMPLAR development; a single-workstation system might be able to plan an ATO, but would be useless in the operational context. The implemented TEMPLAR software incorporates a successful mechanism for distributing the planning process across a network, including the following components:

- A networking backbone capable of linking multiple workstations together and maintaining the Knowledge Base in a consistent state across that network.
- Mechanisms for coordinating the actions of multiple active planners to prevent simultaneous modification of the same Knowledge Base elements.
- Mechanisms for coordinating resource allocation across the network.
- Mechanisms for providing results of planning decisions across the network.

Implementation of TEMPLAR as a distributed network of workstations provides the

ability to allocate system resources to the development of multiple ATO's simultaneously. In addition, the use of multiple distributed workstations applies greater computing power to the planning process, and provides hardware redundancy and sizing flexibility not possible in a centralized system.